x264 for professional video work

An in-depth guide by Space Banana

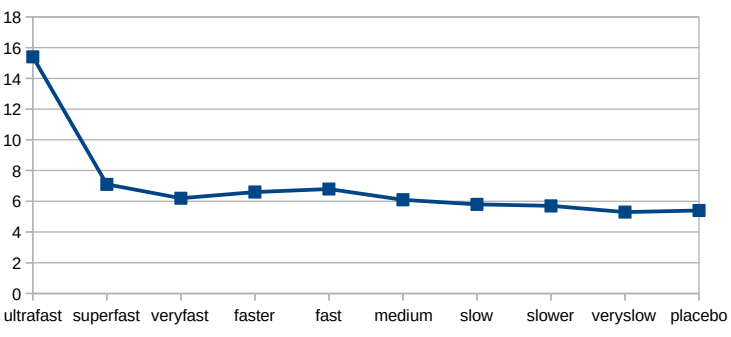
x264 is the most popular and successful encoder for the H.264 video encoding format. It has a very wide feature set and takes advantage of everything H.264 supports, such as lossless compression, zero subsampling, RGB color format, custom keyframe intervals and much more. Besides the feature-set, the encoder is very customizable, with many encoding parameters, but what really sells it for most people is how efficient it is. x264 is a very fast encoder, while also being highly efficient in compression, being able to encode videos at high quality and low bitrates simultaneously. This is a huge deal for delivering videos, especially to the internet. Sure, you could use an H.265 or VP9 encoder instead for even higher compression efficiency, but these are very slow to encode, especially VP9 with its official codec library.

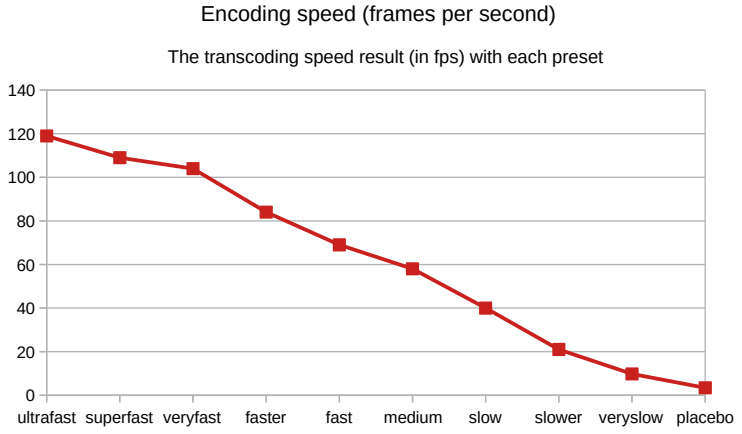
This guide will lead you to how you can professionally use x264 for footage, archiving and editing. The guide focuses on encoding video with FFmpeg and all parameters and arguments mentioned are for FFmpeg.

Encoding performance

Despite its efficient compression, x264 is a very lightweight and optimized encoder. You can use x264 in realtime recordings with a modern budget CPU nowadays, but what if it's still not enough for you? There are multiple ways to optimize your video encoding, but the easiest and most straightforward way to optimize x264 is through choosing a lightweight preset. x264 comes with multiple encoding presets which define encoding parameters to optimize the video encoding for either performance, at the cost of lower compression efficiency and quality, or the opposite.
You can set your preset with the "-preset" or "-preset:v" parameters

The following benchmark uses a CRF-encoded video to determine how compression efficiency and speed vary throughout the presets:





The graph on the left tells us how much the compression efficiency improves throughout the presets. Since the video is CRF-encoded, quality is constant, and bitrate is the one who varies to achieve that quality. We can see that

beyond the veryfast preset there aren't noticeable gains in compression efficiency, while ultrafast is extremely inefficient in comparison.

The graph on the right tells us how heavy these presets are in their encoding speed. As we raise the preset, the encoding slows down almost linearly.

We can conclude that, since speed worsens significantly but compression quality barely changes, that superfast is the most efficient preset when you compare the difference between compression and speed. Veryfast should be used for even better compressions, but it's not recommended to be used in realtime. Ultrafast is ideal for realtime recording due to its performance, but be aware that bitrates will be much higher for the same quality target.

Compression and data preservation

H.264 supports CRF bitrate control and lossless encoding, and so does x264. This makes x264 a very good candidate for producing video with significant quality and data preservation. For more serious workflows, it's advised to use CRF bitrate control rather than CBR.

CBR
CBR stands for constant bitrate. This is the most common and straightforward method of controlling the video bitrate: you set a bitrate target and the video's bitrate will stay in that target the whole time, with just slight fluctiations. Quality differs as the information in the video changes.

CRF
CRF stands for control rate factor. This bitrate control method sets a quality target that the encoder must obey, and bitrate varies unpredictably throughout the video, depending on how much data the video needs to preserve enough of the current information to stay in the determined quality target. Video and encoder parameters also change the behavior if they interfere with the amount of information the video needs.

Unless you have strict storage limits, you should generally use CRF. It guarantees a quality target, regardless of your video and encoding settings. Bitrate differs unpredictability, but so would happen on a lossless video.

Use the argument -crf to determine the CRF value. The accepted value range is 0 to 51. 0 produces lossless video.

Color preservation

Both x264 and H.264 support RGB and YUV at different sampling modes. To preserve colors in video, you should use at least as much sampling as the original footage. If you are recording footage from scratch, you should use as least subsampling as you can. For example, instead of using the color format YUV 4:2:0, you can use 4:2:2 or 4:4:4. You can also encode your video in RGB which doesn't have any subsampling of color, just like YUV 4:4:4.

Fast and lightweight playback

If you are going to use your footage in video editing, VFX and similar tasks, one of the most important factors to keep in mind is how easy and lightweight it is to decode your video. The more lightweight it is, the higher your playback and editing performance. H.264 is a delivery format, and so it doesn't have nearly as much playback performance as more appropriate formats, but you can still optimize your video a lot in x264.

First, you can quickly squeeze more decode performance out of your encodes with the -tune parameter. Use the "-tune fastdecode" argument (this will already be used if your preset is ultrafast).

By default x264 uses interframe compression. In interframe compression, frames share identical pixels' data with each other to optimize the video's file size. This sharing happens in an interval, between what's called keyframes. Keyframes are frames which do not share or are shared data with any other frame.
While this technique is very good to lower bitrate requirements, it's very heavy to decode. Lower the keyframe interval of your encode with the -g parameter. "-g 60" will lower the interval to 60 frames. If you want to fully disable interframe compression, use "-g 0".

Among the frames that share pixel data, we have b-frames. These frames share data for frames both after and before them, and they are also a bottleneck in your playback performance. To disable b-frames, use "-bf 0".